Motion Planning for Steep Hill Climbing

Damion Dunlap, Wei Yu, Emmanuel G. Collins Jr., and Charmane V. Caldwell

***Abstract*— The motors or engines of an autonomous ground vehicles (AGV) have torque and power limitations, which limit their abilities to climb *steep hills*, which are defined to be hills that have high grade sections in which the vehicle is forced to decelerate. Traversal of a steep hill requires the vehicle to have sufficient momentum before entering the hill. This problem is part of a larger class of momentum-based motion planning problems such as the problem of lifting heavy objects with manipulators. Hence, solutions to the steep hill climbing problem have much wider applicability. The motion planning here is accomplished using a dynamic model of the skid-steered AGV used in the experiments along with *Sampling Based Model Predictive Control* (SBMPC), a recently developed input sampling planning algorithm that may be viewed as a generalization of LPA* to the direct use of kinodynamic models. The motion planning is demonstrated experimentally using two scenarios, one in which the robot starts at rest at the bottom of a hill and one in which the robot starts at rest a distance from the hill. The first scenario requires the AGV to first reverse direction so that the vehicle can gather enough momentum before reaching the hill. This corresponds to having the vehicle begin at a local minimum, which results in a problem that many traditional model predictive control methods cannot solve. It is seen that, whereas open loop trajectories can lead to vehicle immobilization, SBMPC successfully uses the information provided by the dynamic model to ensure that the AGV has the requisite momentum.**

I. INTRODUCTION

Recent years have seen a growing interest in the use of dynamic models in motion planning. The concept of kinodynamic planning was initially presented in [1]. The methodologies vary, but are usually based on either the randomized sampling-based approach presented by LaValle and Kuffner in [2] or potential field functions similar to those presented in [3]. This paper describes a particular motion planning problem that benefits greatly from the use of a dynamic model. Particularly, it concerns developing trajectories for climbing *steep hills*, which are defined to be hills that have high grade sections in which the vehicle is forced to decelerate due to the torque and power limitations of the vehicle motors or engine. For problems of this type it is essential that the planner ensures that the vehicle has the necessary momentum to traverse the hill. The required momentum can be computed using a dynamic model of the vehicle that includes the limitations of the vehicle actuation.

The steep hill climbing problem is part of a larger class of motion planning problems that depend upon momentum. For example, the problems of traversing a stretch of viscous mud or traveling over an area of high, stiff vegetation may share these momentum requirements. Another momentum planning problem involves a manipulator lifting a *heavy weight*, defined here to be a weight for which the workspace of the manipulator is limited if the manipulator moves quasi-statically. It follows that solutions to the steep hill climbing problem are expected to find applications in a variety of important motion planning tasks.

This research incorporates a motion planning algorithm called *Sampling Based Model Predictive Control* (SBMPC) [4], [5]. SBMPC allows planning with kinodynamic models and here is used to plan using a relatively complex dynamic model. Like more standard versions of *Model Predictive Control* (MPC), SBMPC simultaneously determines the optimal control input trajectory and the corresponding (kinematically or dynamically feasible) vehicle trajectory. As its name implies, SBMPC depends on sampling, particularly sampling the system inputs, and is essentially an extension of LPA* [6] to kinodynamic models. SBMPC's name derives from the fact that its development was motivated by MPC and it can be used to solve MPC problems with nonlinear models and/or constraints, including problems outside the realm of robotics.

This paper experimentally demonstrates motion planning for steep hill climbing of a skid-steered robot. The hill in these experiments has constant slope, i.e., it is a ramp. In one of the two cases considered, the robot begins at rest at the bottom of the hill and must back up in order to give the vehicle time to accelerate to the momentum needed to traverse the hill. Hence, the robot begins at a local minimum, resulting in an optimization problem which many traditional MPC approaches cannot solve. Due to SBMPC's roots in LPA*, it is able to solve local minimum problems and is shown to successfully compute a trajectory up the hill.

To our knowledge the problem of motion planning for steep hill climbing has not been considered in prior literature. However, the research of [7] considers the problem of using MPC to follow a *given* trajectory up steep slopes. The

D. Dunlap is with the Automation & Dynamics Branch, Littoral Warfare Science & Technology Department of the US Naval Surface Warfare Center Panama City Division, USA damion.d.dunlap@navy.mil

W. Yu and E. Collins are with the Center for Intelligent Systems, Control and Robotics (CISCOR) and the Department of Mechanical Engineering, Florida A&M University-Florida State University, Tallahassee, FL 32310, USA {yuwei, ecollins}@eng.fsu.edu

C. Caldwell is with CISCOR and the Department of Electrical Engineering, Florida A&M University-Florida State University, Tallahassee, FL 32310, USA cvcaldwe@eng.fsu.edu

This work was supported by the U.S. Army Research Laboratory under the Robotics Collaborative Technology Alliance program, Cooperative Agreement W911NF-10-2-0016, and by the National Science Foundation under award CMMI-0927040.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Motion Planning For Steep Hill Climbing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **US Naval Surface Warfare Center,Automation & Dynamics Branch, Littoral Warfare ,Science & Technology Department,Panama City,FL,32407**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the IEEE International Conference on Robotics and Automation, Shanghai, China, May 9-13, 2011

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**9**	

trajectories are planned in advance without an awareness of the actuator limitations and it is hence possible that a given trajectory is not feasible. The results show improvement over using a kinematic model, but are limited to simulations.

The paper is organized as follows. Section II describes the dynamic model used in this research, which was developed in [5] and emphasizes important model features that enable it to be used for motion planning in general and the steep hill climbing problem in particular. Section III provides a brief review of the basic SBMPC algorithm and discusses modifications that make it more efficient for problems of the type considered here. Section IV presents experimental results that demonstrate that trajectories generated without taking into account the actuator limitations may result in vehicle immobilization, while SBMPC is able to generate feasible trajectories. Finally, Section V presents conclusions and future work.

II. DYNAMIC MODEL FOR 3D HILL CLIMBING

This section describes the dynamic model of a skid-steered wheeled vehicle that was developed and experimentally verified in [8]. The presentation emphasizes the importance of using a closed-loop model (i.e., one that includes the motor speed controllers) to reduce the uncertainty of the tire/ground interaction, the inclusion of the motor limitations, and the ability of the model to predict deceleration when climbing a steep hill.

A. Dynamic Model for 3D Linear Motion

Fig. 1 is the free body diagram of a skid-steered wheeled vehicle. It is assumed that the surface elevation is described by $Z = f(Y)$ such that the left and right front wheels experience the same elevation and likewise for the rear wheels. Let β denote the angle between the global coordinate axis Y and the body-fixed axis y_r (which can be determined analytically from $Z = f(Y)$), W the weight of vehicle, f_{r_r} the rolling resistance of the right wheels, and F_r the traction force that acts on the vehicle. The left wheel forces are identical to those of the right wheels and are not shown in Fig. 1.

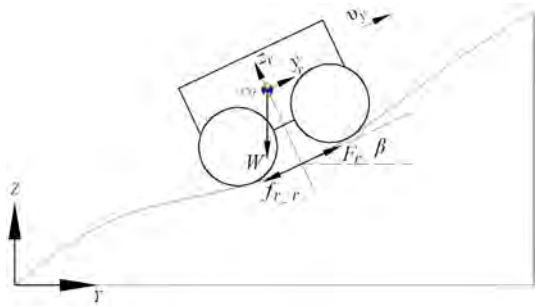

Fig. 1. Free-body diagram for vehicle hill climbing

The 3D dynamic model for linear hill climbing [8] is given by

$$\begin{bmatrix} \frac{mr^2}{4}+\frac{r^2 I}{\alpha B^2} & \frac{mr^2}{4}-\frac{r^2 I}{\alpha B^2} \\ \frac{mr^2}{4}-\frac{r^2 I}{\alpha B^2} & \frac{mr^2}{4}+\frac{r^2 I}{\alpha B^2} \end{bmatrix} \ddot{q} + \begin{bmatrix} \tau_{l\_Res} \\ \tau_{r\_Res} \end{bmatrix} + \begin{bmatrix} \frac{mgr\sin\beta}{2} \\ \frac{mgr\sin\beta}{2} \end{bmatrix} = \begin{bmatrix} \tau_l \\ \tau_r \end{bmatrix}, \quad (1)$$

where m and I are respectively the mass and moment of inertia of the vehicle, α is a terrain-dependent parameter ($= 1$ for linear motion as assumed here), B is the vehicle width, r is the wheel radius, $q = [\theta_l \ \theta_r]^T$ is the angular displacement of the left and right wheels, $\tau_{l\_Res} = r f_{l\_r}$ is the resistance torque of the left wheel, $\tau_{r\_Res} = r f_{r\_r}$ is the resistance torque of the right wheel, $\tau_l = r F_l$ is the applied torque of the left motor, and $\tau_r = r F_r$ is the applied torque of the right motor.

B. Closed-loop Control System

The dynamic model (1) is an essential part of the prediction model used for motion planning. However, no matter how accurate the analysis, the model still has uncertain parameters, e.g., the coefficient of rolling resistance. In this research, the closed-loop control model is utilized to predict motion as shown in Fig. 2, since the feedback system can reduce the effects of the model uncertainty [8]. The PID controller is used for speed control of the left or right side wheels, and the vehicle dynamics and the interaction of the vehicle with the terrain are described by (1).

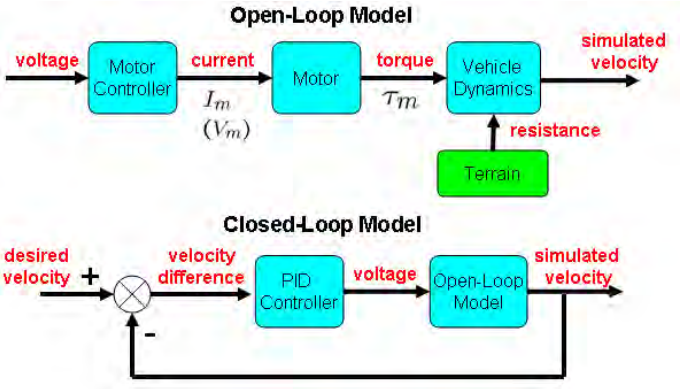

Fig. 2. The closed-loop control system for the left or right side of a skid-steered wheeled vehicle.

1) Modeling the Motor Limitations: The motor controller block outputs the current I_m to the motor and was configured to enforce the current constraint,

$$I_m < I_{max}, \quad (2)$$

where I_{max} represents the maximum current allowable before the motor is in danger of overheating. Also, for the Maxon 4-Q-DC motor controller used in this research the maximum output voltage $V_{m,max}$ varies with the current I_m and is governed by

$$V_{m\_max} = PWM \cdot (V_{CC} - U_{loss}) - \frac{\frac{\Delta n_m}{\Delta \tau_m} K_T I_m}{K_n}, \quad (3)$$

where PWM is the pulse-width modulation maximum duty cycle, V_{CC} is the battery supply voltage, U_{loss} is the voltage loss in the motor controller, $\Delta n_m/\Delta\tau_m$ is the speed-torque gradient of the motor, and K_n is the speed constant.

The motor block included the speed vs. current curve for a DC motor [9], which is of the form

$$\omega_m = -aI_m + b(V_m), \tag{4}$$

where ω_m is the angular velocity of the motor, $a > 0$ is a constant motor parameter, and $b(V_m)$ is a function that changes monotonically with the motor voltage V_m.

Due to the current constraint (2) of the motor block, the torque τ_m for the motors on each side of the vehicle is constrained by

$$\tau_m \leq \tau_{max} = K_T I_{max} g_r \eta, \tag{5}$$

where K_T is the torque constant, g_r is the gear ratio and η is the efficiency. The power P_m of the motors on each side of the vehicle is constrained by

$$\tau_m \omega_m = P_m \leq P_{max} = V_{m\_max} I_m, \tag{6}$$

where V_{m_max} is given by (3). Note that P_{max} is not constant, but varies with the current I_m.

In the research platform of Fig. 8 the DC motor is a Pittman GM 9236. Based on the specifications for the motor, $I_{max} = 5.5$ A. By employing the appropriate numerical values for the parameters in (3), (6) results in the P_{max} vs. I_m curve of Fig. 3. The square symbol in Fig. 3 represents the largest possible output power for one side of the vehicle, which is 51 W.

Generally speaking, the torque limit constraint (5) is what causes deceleration when climbing a steep hill, while the power constraint (6) limits the speed of the vehicle while traveling on either horizontal or sloped terrains. The speed limitations are expected to be particularly important when planning minimum time paths on undulating terrain.

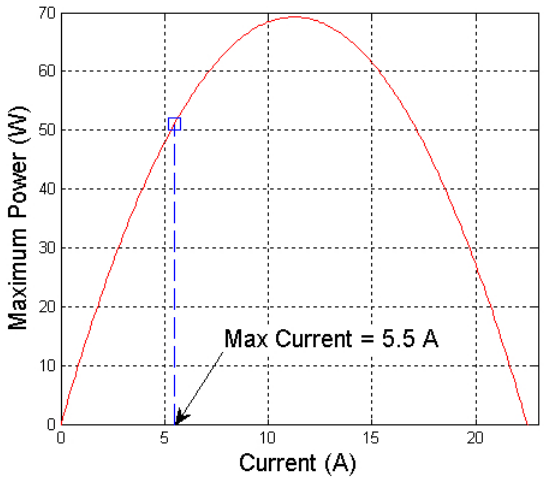

Fig. 3. Maximum power vs. current for the motor controller.

2) Comparison of Open-Loop and Closed-Loop Modeling: Fig. 4 shows comparisons of both the closed-loop and open-loop experimental and simulation velocity results for the modified Pioneer 3-AT[1] shown in Fig. 8 and demonstrates the advantage of using the closed-loop system for velocity prediction instead of the open-loop system. The vehicle is commanded at an acceleration of 1 m/s^2 to a velocity of 0.2 m/s for straight-line movement on the lab vinyl surface. Predictions using the closed-loop model closely match the experimental results, whereas when the torques from the closed-loop experiment were used to drive the open-loop model, the prediction error is significant and increases over time. Hence, the closed-loop model is used here for motion planning.

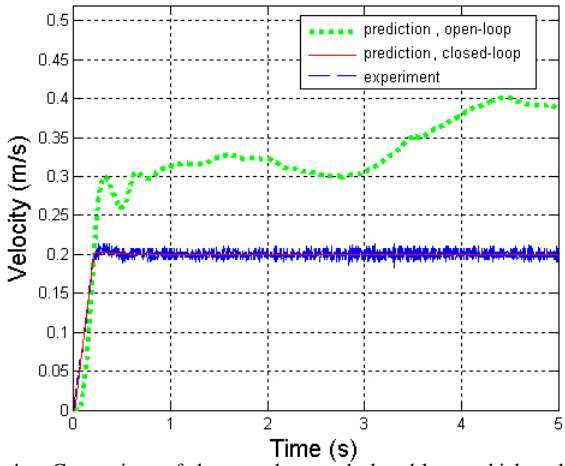

Fig. 4. Comparison of the open-loop and closed-loop vehicle velocities when the vehicle is commanded to 0.2 m/s for straight-line movement on the lab vinyl surface

C. Ability of the Closed-Loop Model to Predict Deceleration

The closed-loop model was used to predict the behavior of the vehicle when traversing hills of constant slope (i.e., β in Fig. 1 is constant). These simulated predictions were compared with the experimental results. There was generally a close correlation between the simulation and experimental results, an illustration of which is given in Fig. 5. This figure shows the ability of the closed-loop model to predict the deceleration due to the motor torques saturating at 6.2 Nm.

III. SAMPLING BASED MODEL PREDICTIVE CONTROL

SBMPC is a novel approach that allows real time motion planning that uses the vehicle's nonlinear model and avoids local minima. The method employs an MPC type cost function and optimizes the inputs, which is standard in the control community. Instead of using traditional numerical optimization, SBMPC applies sampling and a goal-directed (A^*-type) optimization, which are standard in the robotic and AI communities. This section provides a brief overview of the methods from which SBMPC was derived. It then

[1]This Pioneer 3-AT contains a control system designed and implemented in the research lab rather than the original control system.

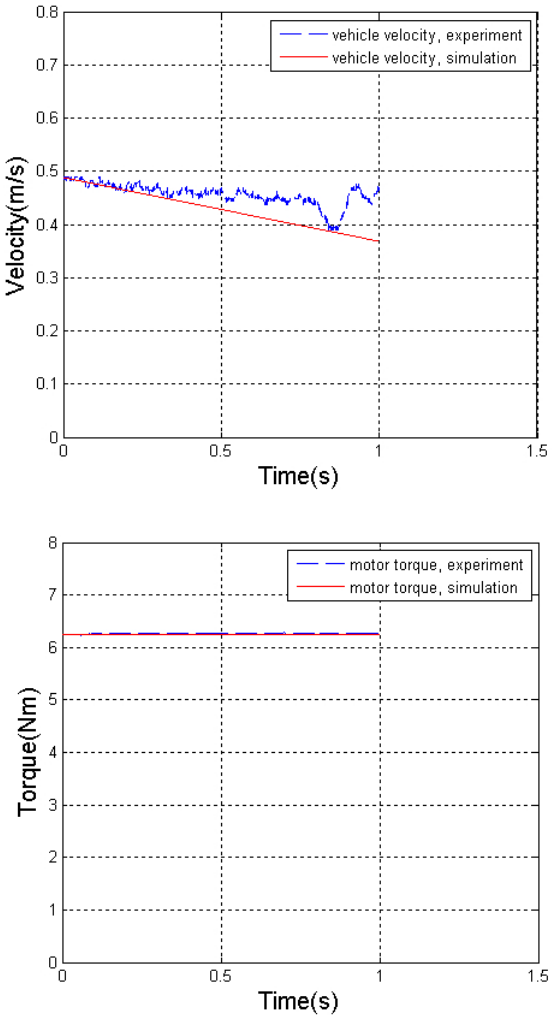

Fig. 5. Closed-loop vehicle velocity and torque comparison when commanded linear velocity=1.2 m/s with 0.49m/s initial velocity for wood-board hill climbing with slope $\beta = 15.0°$

describes the SBMPC cost function and outlines the SBMPC algorithm. Next, it disusses the benefits of SBMPC. Finally, it describes how SBMPC was specialized to the steep hill climbing problem. This brief presentation of the SBMPC algorithm does not attempt to completely describe the algorithm but to give an overview of the approach that focuses on the principles upon which the algorithm was developed. The specifics of the algorithm are presented in [5].

A. Model Predictive Control

Introduced to the process industry in the late 1970s [10], Model Predictive Control (MPC) is a mixture of system theory and optimization. It is a control method that finds the control input by optimizing a cost function subject to constraints. The cost function calculates the desired control signal by using a model of the plant to predict future plant outputs. MPC generally works by solving an optimization problem at every time step k to determine control inputs for the next N steps, known as the prediction horizon. This optimal control sequence is determined by using the system model to predict the potential system response, which is then evaluated by the cost function J. Most commonly, a quadratic cost function minimizes control effort as well as the error between the predicted trajectory and the reference trajectory r. The prediction and optimization operate together to generate sequences of the controller output u and the resulting system output y. In particular, the optimization problem is

$$\min\ J = \sum_{i=1}^{N} \|r(k+i) - y(k+i)\|_Q^2 + \sum_{i=0}^{M-1} \|\Delta u(k+i)\|_S^2 \tag{7}$$

subject to the model constraints,

$$x(k+i) = f\left(x(k+i-1), u(k+i-1)\right) \tag{8}$$

$$y(k+i) = g\left(x(k+i)\right) \tag{9}$$

and the inequality constraints,

$$\begin{gathered} Ax \leq b \\ C(x) \leq 0 \\ u^l \leq u(k+i) \leq u^u \end{gathered} \tag{10}$$

Traditional MPC has typically been computationally slow and often incorporates only simple linear models.

B. Sampling Based Motion Planning

Sampling-based motion planning algorithms include Rapidly-exploring Random Tree (RRTs) [11], probability roadmaps [12], and randomized $A^\star$ algorithms [13]. A common feature of each of those algorithms to date is that they work in the output space of the robot and employ various strategies for generating samples (i.e., random or pseudo-random points). In essence, as shown in Fig. 6, sampling-based motion planning methods work by using sampling to construct a tree that connects the root with a goal region. The general purpose of sampling is to cover the space so that the samples are uniformly distributed, while minimizing gaps and clusters [14].

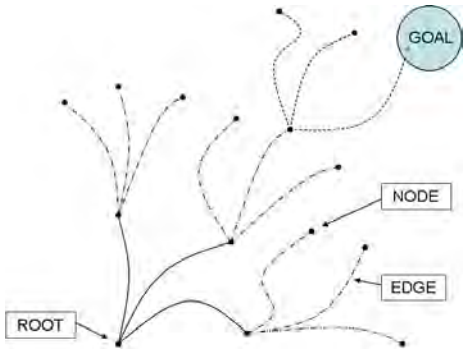

Fig. 6. A tree that connects the root with a goal region.

C. Goal Directed Optimization

There is a class of discrete optimization techniques that have their origin in graph theory and have been further developed in the path planning literature. In this paper these techniques will be called *goal-directed optimization* and refer

to graph search algorithms such as Dijkstra's algorithm and the $A^\star$, $D^\star$, and $LPA^\star$ algorithms [15], [16]. Given a graph, these algorithms find a path that optimizes some cost of moving from a start node to some given goal. Although not commonly recognized, goal-directed optimization approaches are capable of solving control problems for which the ultimate objective is to generate an optimal trajectory and control inputs to reach a goal (or set point) while optimizing a cost function; hence, they apply to terminal constraint optimization problems and set point control problems.

D. The SBMPC Optimization Problem

SBMPC overcomes some of the shortcomings of traditional MPC by sampling the input space as opposed to sampling the output space as in traditional sampling-based motion planning methods. The need for a nearest-neighbor search is eliminated and the local planning method (LPM) is reduced to the integration a system model and therefore only generates outputs that are achievable by the system. To understand the relationship between sampling-based algorithms and MPC optimization, it is essential to pose sampling-based motion planning as an optimization problem. To illustrate this point, note that *, subject to the constraints of the sampling,* a goal-directed optimization algorithm can effectively solve the mixed integer nonlinear optimization problem,

$$\min_{\{u(k),\ldots,u(k+N-1)\},N} J = \sum_{i=0}^{N} \parallel y(k+i+1) - y(k+i) \parallel_{Q(i)} + \sum_{i=0}^{N-1} \parallel \Delta u(k+i) \parallel_{S(i)} \quad (11)$$

subject to the system equations,

$$x(k+i) = f(x(k+i-1), u(k+i-1)), \quad (12)$$

$$y(k) = g(x(k)), \quad (13)$$

and the constraints,

$$\parallel y(k+N) - \mathbf{G} \parallel \leq \epsilon, \quad (14)$$

$$x(k+i) \in \mathbf{X}_{free} \ \forall \ \ i \leq N, \quad (15)$$

$$u(k+i) \in \mathbf{U}_{free} \ \forall \ \ i \leq N, \quad (16)$$

where $\Delta u(k+i) = u(k+i) - u(k+i-1)$, $Q(i) \geq 0$, $S(i) \geq 0$, and $\mathbf{X}_{free}$ and $\mathbf{U}_{free}$ represent the states and inputs respectively that do not violate any of the problem constraints. The term $\parallel \ y(k+i+1) - y(k+i) \ \parallel_{Q(i)} + \parallel \ \Delta u(k+i) \ \parallel_{S(i)}$ represents the edge cost of the path between the current predicted output $y(k+i)$ and the next predicted output $y(k+i+1)$. The goal state G is represented as a terminal constraint as opposed to being explicitly incorporated into the cost function. Goal-directed optimization methods implicitly consider the goal through the use of a function that computes a rigorous lower bound of the cost from a particular state to G. This function, often referred to as an "optimistic heuristic" in the robotics literature, is eventually replaced by actual cost values based on the predictions and therefore does not appear in the final cost function. The cost function can be modified to minimize any metric as long as it can be computed as the sum of edge costs.

E. The SBMPC ALGORITHM

The formal SBMPC algorithm can be found in [5]. However, the main component of the SBMPC algorithm is the optimization, which will be called Sampling-Based Model Predictive Optimization and consists of the following steps:

1) **Sample Control Space:** Generate a set of samples of the control space that satisfy the input constraints.
2) **Generate Neighbor Nodes:** Integrate the system model with the control samples to determine the neighbors of the current node.
3) **Evaluate Node Costs:** Use an A^*-like heuristic to evaluate the cost of the generated nodes based on the desired objective (shortest distance, shortest time, or least amount of energy, etc.).
4) **Select Lowest Cost Node:** The nodes are collected in the Open List, which ranks the potential expansion nodes by their cost. The Open List is implemented as a heap so that the lowest cost node that has not been expanded is on top.
5) **Evaluate Edge Cost for the "Best" Node:** Evaluate each of the inequality constraints described in (6) for the edge connecting the "best" node to the current node. The edge cost evaluation requires sub-sampling and iteration of the model with a smaller time step for increased accuracy; it is therefore only computed for the current "best"node. In the worst case the edge cost of all of the neighbor nodes will be evaluated, which is how A^* typically computes cost.
6) **Check for Constraint Violations:** If a constraint violation occurred, go back to step 4 and get the next "best" node.
7) **Check for Completion:** Determine if the current solution contains a path to the goal. If yes, stop. If no, go back to step 1.

The entire algorithm is integrated into the MPC framework by executing the first control and repeating the optimization until the goal is reached since the completion of SBMPO represents the calculation of a path to the goal and not the complete traversal.

F. Benefits of SBMPC

The SBMPC approach has several benefits. First, SBMPC is a method that can address problems with nonlinear models. It effectively reduces the problem size of MPC by sampling the inputs of the system, which can considerably reduce the computation time. In addition, the method also replaces the traditional MPC optimization phase with $LPA^\star$, an algorithm derived from $A^\star$ that can replan quickly (i.e., it is incremental). SBMPC retains the computational efficiency and has the convergence properties of $LPA^\star$ [16], while avoiding some of the computational bottlenecks associated with sampling-based motion planners.

G. SBMPC Hill Climbing

The goal was to apply SBMPC to the hill climbing problem in a computationally efficient manner. The model input was the commanded (or desired) vehicle acceleration, which was assumed to lie in the range $[-a_{max}\ a_{max}]$. Hence, SBMPC sampled the desired vehicle accelerations and the desired velocity of the closed-loop model of Fig. 2 varied linearly over each sample period with the slope being the sampled acceleration. Since SBMPC used a fairly complex dynamic model, it was desired to minimize the number of times the model was integrated. Although the cost function was chosen to represent the distance traveled by the vehicle, internally SBMPC was modified so that its ultimate objective was to achieve a vehicle velocity at the base of the hill that yielded at least the minimum momentum needed to traverse the hill. This was accomplished by enforcing that once a trajectory being considered by SBMPC reached the bottom of the hill, the vehicle would be commanded with a_{max} until it either reached the hill top or failed by coming to a zero or negative velocity before reaching the hill top. This modification of SBMPC enabled it to avoid looking at the potentially large number of trajectories that would result from continuing to sample the commanded acceleration as the vehicle traversed the hill.

Assume that a trajectory was unable to reach the hill top and had a velocity v_b at the bottom of the hill. Then, as shown in Fig. 7, a virtual obstacle was imposed in the Position-Velocity space to ensure that only trajectories with a hill base velocity greater than that of the unsucessful trajectory would be considered in later iterations of SBMPC. Note that this virtual obstacle considers not only the velocity at the base of the hill, but also all position-velocity combinations that appear prior to the base and cannot achieve the desired base velocity even if the acceleration maintains its maximum (commanded) value a_{max}. This adaptation allowed SBMPC to utilize the problem physics to prevent computations of trajectories that will definitely be infeasible based on knowledge gained in a prior algorithm iteration.

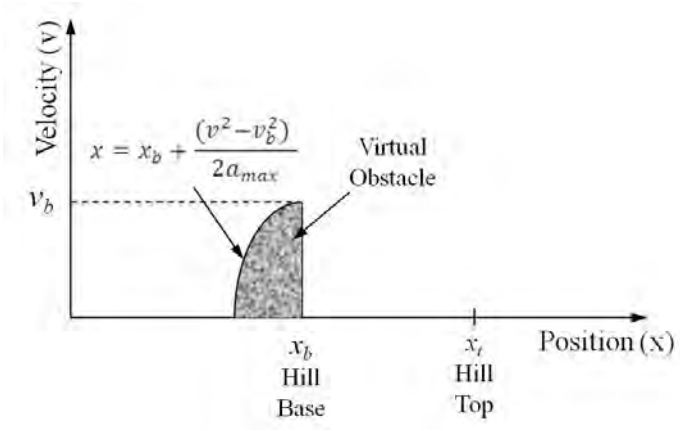

Fig. 7. Virtual Obstacle used in Position-Velocity space for SBMPC hill climbing

IV. EXPERIMENTAL RESULTS

This section describes motion planning for linear hill climbing using the experimentally verified closed-loop model of Section II-B. The plans were generated off line by the modified SBMPC algorithm described in Section III.

Fig. 8 shows a modified Pioneer 3-AT at the bottom of a hill attempting to climb the hill. Two hill climbing scenarios are considered below. In Scenario 1 the robot starts from rest at the bottom of the hill (the position shown in Fig. 8). In Scenario 2 the robot starts from rest 1 m away from the bottom of the hill as shown in Fig. 9. Scenario 1 corresponds to a situation in which the robot is required to reverse direction in order to move forward and hence tests the ability of SBMPC to escape a local minimum. Scenario 2 is a more standard situation in which the robot is required to simply speed up to the required momentum in order to traverse the hill. For both cases it is experimentally demonstrated that a trajectory generated by the use of SBMPC with the closed-loop dynamic model of Pioneer 3-AT enables the robot to climb to the top of the hill, while a trajectory generated without taking into the vehicle dynamics leads to the inability of the robot to climb to the top of the hill.

Fig. 8. The modified Pioneer 3-AT at the bottom of a hill

Fig. 9. The modified Pioneer 3-AT 1 meter from the bottom of a hill

The closed-loop model of Fig. 2 has as its input the commanded vehicle velocity profile. In these experiments, SBMPC sampled the desired acceleration in the range $[-1\ 1]$ m/s^2. Hence, the desired vehicle velocity always varied linearly with slope equal to the sampled acceleration.

For Scenario 1 the vehicle had the initial commanded velocity shown in Fig. 10. After approximately 5.5 s, the robot's actual velocity was 0 m/s, which resulted in the robot being immobilized in the middle of the hill. Fig. 11 shows the result for hill climbing using SBMPC, which commanded the robot to back up and then accelerate to a velocity of 0.55 m/s at 1.5 s, a velocity maintained until approximately 2.3 s, the time at which the vehicle was positioned at the bottom of the hill. This commanded velocity profile resulted in the vehicle's front wheels reaching the top of the hill at approximately 4.1 s. A time-lapse sequence of the motion with and without SBMPC is shown in Figure 12. SBMPC took 0.180 s to compute the implemented trajectory on a 2.4GHz intel core 2 duo processor.

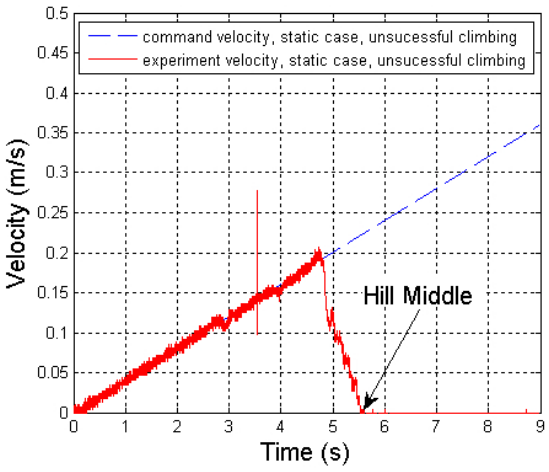

Fig. 10. Commanded vs. experimental velocity for Scenario 1 without using SBMPC

For Scenario 2 the vehicle had the initial commanded velocity shown in Fig. 13. After approximately 6.3 s, the robot's actual velocity was 0 m/s, which resulted in the robot being immobilized before reaching the top of the hill. Fig. 14 shows the result for hill climbing using SBMPC, which commanded the robot to accelerate to a velocity of 0.55 m/s at 3 s, the time at which the vehicle was positioned at the bottom of the hill. As the robot climbed the hill, it decelerated, resulting in a continual decrease in velocity. When the vehicle's front wheels reached the top of the hill (at 4.7 s), the robot again accelerated to the commanded velocity. Hence, the commanded velocity profile resulted in the vehicle reaching the top of the hill. SBMPC took 0.003 s to compute the implemented trajectory on a 2.4GHz intel core 2 duo processor.

V. CONCLUSION

This paper provided experimental demonstration of steep hill climbing for an AGV using Sampling Based Model

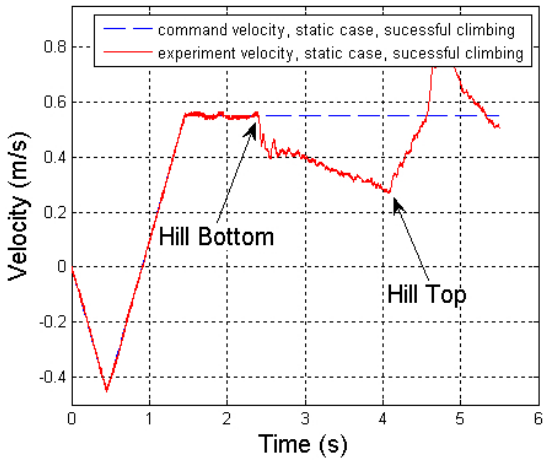

Fig. 11. Commanded vs. experimental velocity for Scenario 1 using SBMPC

Predictive Control (SBMPC) in conjunction with a (closed-loop) dynamic model that includes the actuator limitations in addition to the internal control systems (i.e., the speed controllers for the motors) to reduce the model uncertainty, most notably the uncertainty associated with the vehicle/ground interaction. SBMPC was seen to have the ability to escape the local minimum associated with one of the two scenarios and also enabled the development of smooth trajectories, specifically trajectories for which the commanded velocity is continuous. It is conjectured here that properly designed input sampling methodologies will be found to be more natural and efficient kinodynamic planning algorithms as the dynamic models increase in complexity.

To adopt SBMPC to the steep hill climbing problem, it was required to recognize a steep hill as a potential "traversable obstacle." This requires information on the grade of slopes for a given terrain type that identify a steep hill and intelligent perception algorithms that can reliably recognize these slopes. The slope grades can be determine by off-line analysis and simulation using the dynamic model. The perception problem was not considered here. It was simply assumed that the vehicle knew the hill slope.

The motion planning research will be expanded in future work. First, the SBMPC algorithm has been experimentally applied to develop near minimum time trajectories that have the necessary momentum to lift heavy loads for a single link manipulator. The efficient computation of trajectories that terminate with zero velocity at the desired goal position is enabled by the development of an optimistic $A^\star$ heuristic based on the solution of a minimum time control problem for the system $\ddot{q} = u$, where $u_{min} < u < u_{max}$. This ongoing work will be reported in a future publication. The general approach can be used to specify the vehicle velocity at the top of the hill in the steep hill climbing problem. Second, it is important to consider curvilinear hill climbing. It was not considered in the present research primarily because the corresponding dynamic model does not exist for skid-

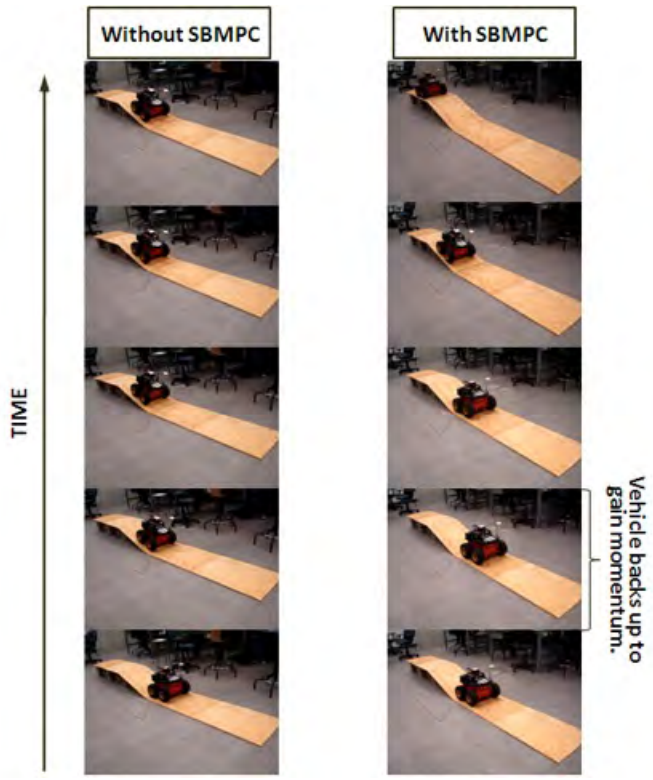

Fig. 12. Hill climbing time-lapse sequence for Scenario 1

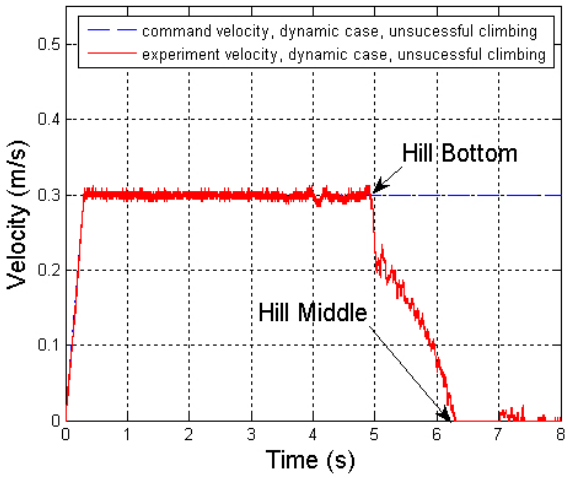

Fig. 13. Commanded vs. experimental velocity for Scenario 2 without using SBMPC

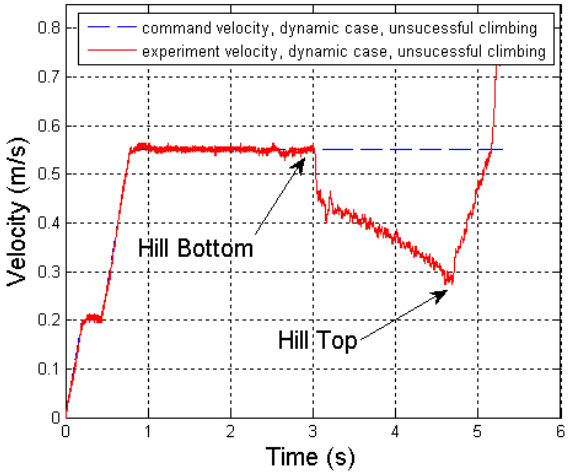

Fig. 14. Commanded vs. experimental velocity for Scenario 2 using SBMPC

steered vehicles. However, the appropriate models are under development. The related problems of traversing mud and high, stiff vegetation are also of interest with the main issue being a technique for effective characterization of the vehicle-ground interaction.

References

[1] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Jounal of the Assocation for Computing Machinery*, 40(5):1048–1066, November 1993.

[2] S. M. LaValle and J. M. Kuffner Jr. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[3] A. A. Masoud. Kinodynamic motion planning: A novel type of nonlinear, passive damping forces and advantages. *IEEE Robotics & Automation Magazine*, pages 85–99, March 2010.

[4] D. D. Dunlap, E. G. Collins, Jr., and C. V. Caldwell. Sampling based model predictive control with application to autonomous vehicle guidance. *Florida Conference on Recent Advances in Robotics*, May 2008.

[5] D.D. Dunlap, C.V. Caldwell, and E.G. Collins. Nonlinear model predictive control using sampling and goal-directed optimization. *IEEE Multi-conference on Systems and Control*, September 2010.

[6] S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning A*. *Artificial Intelligence*, 155:93–146, 2004.

[7] S. C. Peters and K. Iagnemma. Mobile robot path tracking of agressive maneuvers on sloped terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 242–247, Nice, France, September 2008.

[8] W. Yu, O. Y. Chuy, Jr. E. G. Collins, and P. Hollis. Analysis and experimental verification for dynamic modeling of a skid steered vehicle. *IEEE Transactions on Robotics*, 26(2):340–353, April 2010.

[9] Giorgio Rizzoni. *Principles and Applications of Electrical Engineering*. McGraw-Hill, 2000.

[10] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Haslow, UK, 2002.

[11] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

[12] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, 12(4):566 – 580, June 1996.

[13] Maxim Likhachev and Anthony Stentz. R* search. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1–7, Apr 2008.

[14] Stephen R. Lindemann and Steven M. LaValle. Incremental low-discrepancy lattice methods for motion planning. *International Conference on Robotics & Automation*, pages 2920–2927, September 2003.

[15] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[16] S Koenig, M Likhachev, and D Furcy. Lifelong planning A*. *Artificial Intelligence*, 2004.